Brownian Motion

Math 365

April 19, 2022

Submit your completed lab to Gradescope.

In this lab we will explore the properties of Brownian motion.

Random Walks to Brownian Motion

Brownian motion can be constructed as the limit of random walks, where we shrink time and spatial steps in concert: $\Delta x = \pm\sqrt{\Delta t}$ at each discrete jump. Let's do a few simulations of random walks, each time shrinking the time step, while keeping the time interval $[0, 1]$ fixed.

Exercise 1

Use the R code below to run several different random walks. Try $N = 10$, 100, 1000, and 10000 (called numsteps in the code) to see what happens qualitatively as the time step shrinks toward 0.

First set up the basic parameters for the random walk simulation:

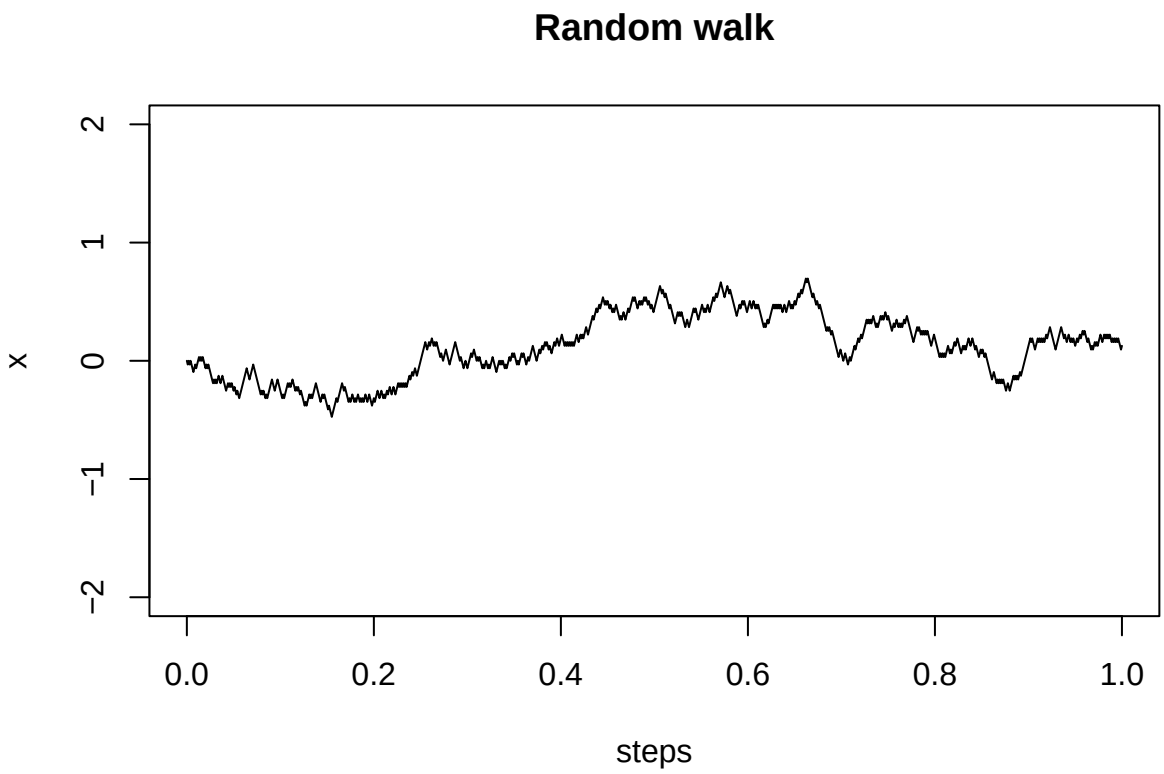
```
duration <- 1  # total time
numsteps <- 1000 # number of steps
steps <- seq(0,1,duration/numsteps)
```

Do a sequence of coin flips to determine whether to go up or down at each step, with each change of position equal to $\pm\frac{1}{\sqrt{N}}$:

```
deltax <- sample(c(-1,1),size=numsteps,replace=TRUE)/sqrt(numsteps)
```

Cumulatively sum these changes in position, assuming $B_0 = 0$, to obtain the random walk:

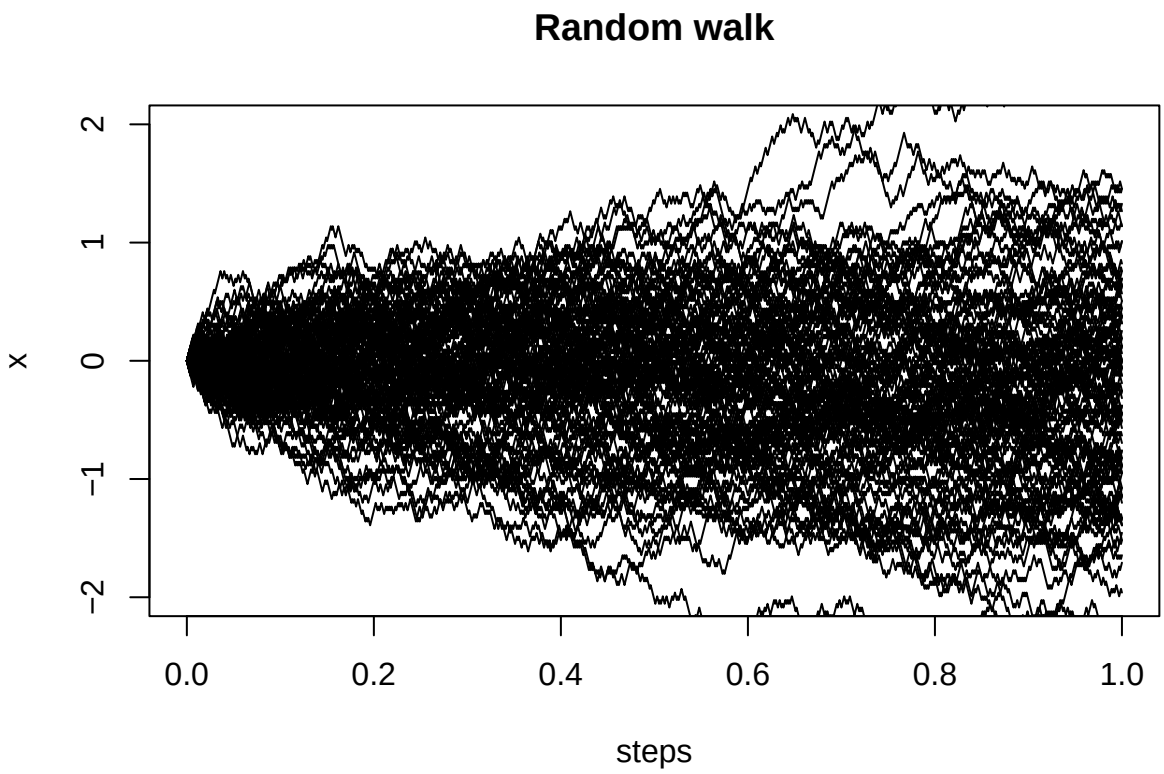
```
x <- c(0, cumsum(deltax)) # compute cumulative sum
plot(steps, x, type = "l", ylim = c(-2, 2), main = "Random walk")
```



Exercise 2

Examine the variability among random walk simulations. Fix the number of steps, e.g., $N = 1000$, and run 100 simulations all plotted together on the same graph. A simple way to do this is to use a for-loop on the code above (but use lines instead of plot to put all on the same graph).

```
plot(steps, x, type = "l", ylim = c(-2, 2), main = "Random walk")
for (k in 2:100){
        deltax <- sample(c(-1,1),size=numsteps,replace=TRUE)/sqrt(numsteps)
        x <- c(0, cumsum(deltax)) # compute cumulative sum
        lines(steps, x, type = "l", ylim = c(-2, 2), main = "Random walk")
}
```



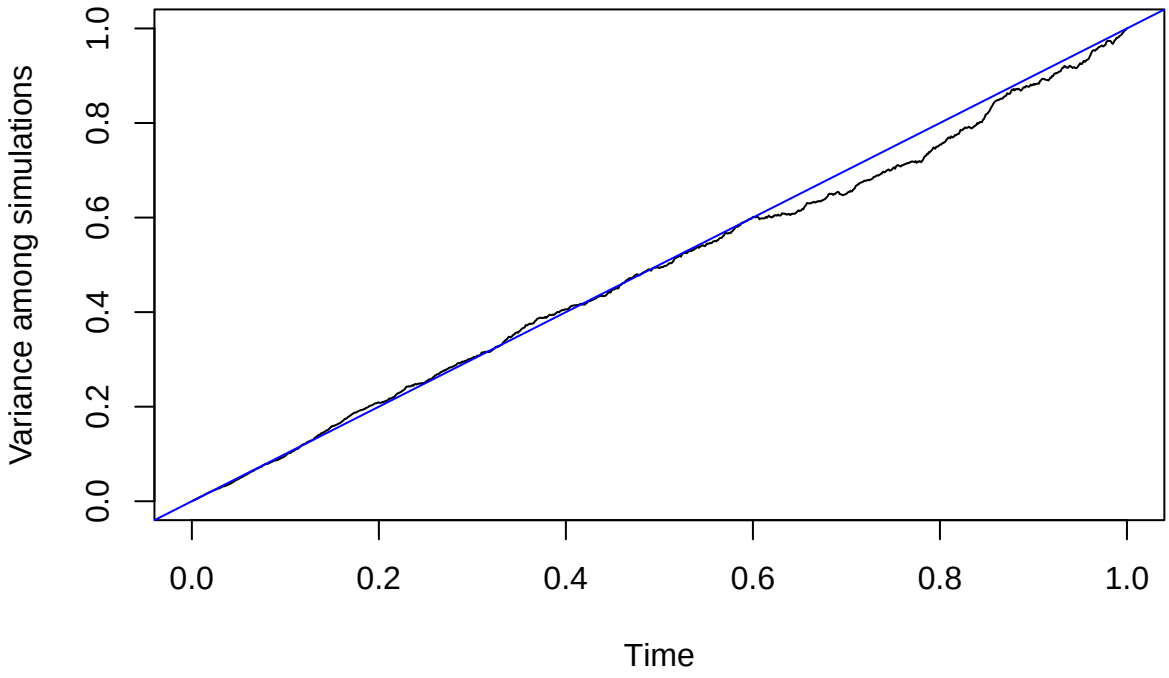
Exercise 3

We can measure how the spread among walks changes over time by storing a set of simulations and calculating the variance of B_t across the instances for each fixed t. The following code stores 1,000 simulations in a matrix:

```
numsim <- 1000
deltaX <- matrix(sample(c(-1,1),size=numsteps*numsim,replace=TRUE)/sqrt(numsteps), numsim, numsteps)
X <- cbind(rep(0, numsim), t(apply(deltaX, 1, cumsum)))
```

To calculate the variance at each time point, apply var to each column of the matrix and then plot:

```
v <- apply(X, 2, var)
plot(steps, v, type = "l", xlab = "Time", ylab = "Variance among simulations")
abline(0,1,col="blue") # theoretical value of the variance (linear with slope sigma2=1)
```

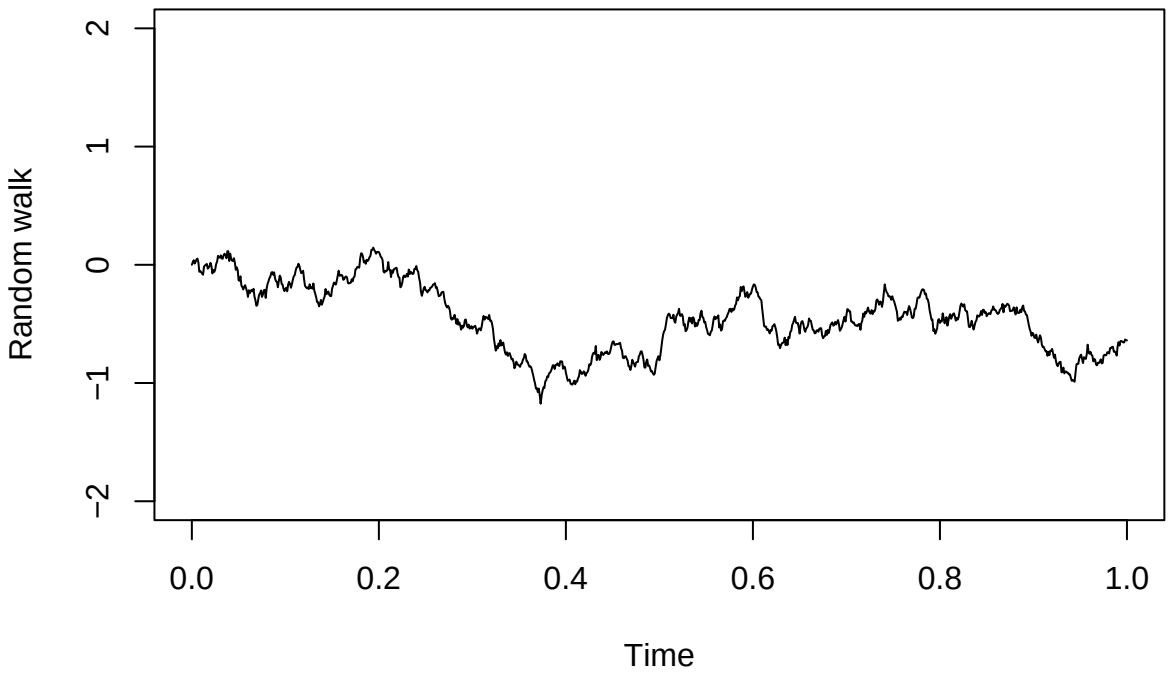


Exercise 4

We can derive a more refined simulation approach by using the theory of Brownian motion. We know the variance of $\Delta x = B_t - B_s$ is $(t-s)\sigma^2$ (length of the time step times the variance parameter). If we divide $[0,1]$ into N equal subintervals, then $\Delta t = \frac{1}{N}$ and the variance of Δx is σ^2/N. Update the previous code by altering how the changes in position are generated:

```
sigma2<-1
deltax <- rnorm(n = numsteps, sd = sqrt(sigma2/numsteps)) # random changes in position

x <- c(0, cumsum(deltax))
plot(steps, x, type = "l", xlab = "Time", ylim = c(-2,2), ylab = "Random walk")
```



Run simulations using this alternative approach and compare to the random walk. Also explore changing the variance parameter σ^2 to see how that effects the motion, e.g., some smaller values like 1/2 and some larger values like 2. What changes qualitatively when the variance parameter changes?

Ans: As we increase the variance, we notice more jaggedness in the behaviour of the random walk graph. While, when we decrease the variance it is notable that the random walk is less jagged.

Exercise 5

Brownian motion with drift can be modeled by adding an underlying linear trend with slope μ to a Brownian motion:

$$Y_t = B_t + \mu t.$$

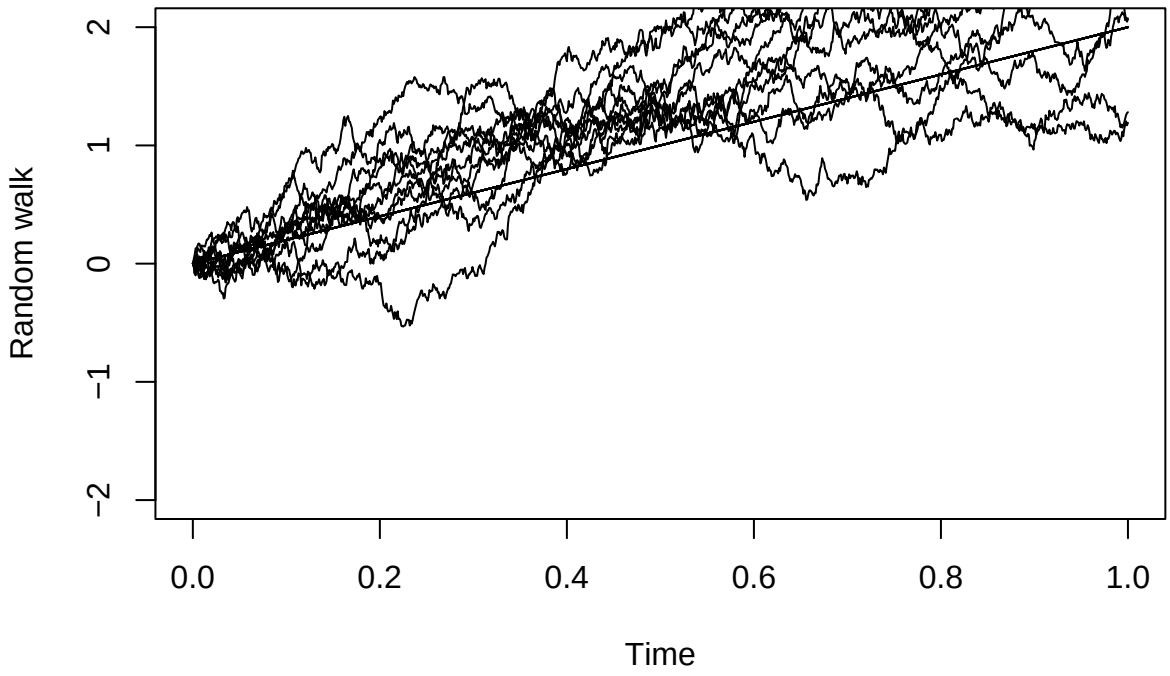
Run simulations of Brownian motion with drift with variance parameter σ^2 and slope μ of your choice. Plot 10 such paths on the same graph (all with the same parameters) along with the trend line.

```
sigma2<-1
        deltax <- rnorm(n = numsteps, sd = sqrt(sigma2/numsteps)) # random changes in position

        x <- c(0, cumsum(deltax))
        plot(steps, x+2*steps, type = "l", xlab = "Time", ylim = c(-2,2), ylab = "Random walk")
        lines(steps, 2*steps)
for (k in 1:10) {
        sigma2<-1
        deltax <- rnorm(n = numsteps, sd = sqrt(sigma2/numsteps)) # random changes in position

        x <- c(0, cumsum(deltax))
```

```
        lines(steps, x+2*steps, type = "l", xlab = "Time", ylim = c(-2,2), ylab = "Random walk")
        lines(steps, 2*steps)
}
```



Exercise 6

Geometric Brownian motion is defined to be

$$Y_t = e^{B_t}$$

where B_t is Brownian motion (could be with drift if desired). Geometric Brownian motion is sometimes used to model stock prices over time, if it appears that the percentage changes are independent and identically distributed. Here's a rough sketch of the idea: Suppose Y_t is the price of some stock at time t and you believe Y_t/Y_{t-1} are i.i.d. for all positive integer values of t. Let $Z_t = Y_t/Y_{t-1}$. Then

$$Y_n = Z_n Y_{n-1} = Z_n Z_{n-1} Y_{n-2} = \cdots = Z_n Z_{n-1} \cdots Z_1 Y_0.$$

Applying a logarithm yields

$$\log(Y_n) = \log(Y_0) + \sum_{i=1}^{n} \log(Z_i),$$

so $\log(Y_n)$ is the sum of i.i.d. random variables (each giving an incremental change in position) and so can be approximated as a Brownian motion. Hence we set $\log(Y_t) = B_t$, where B_t is a Brownian motion, so that $Y_t = e^{B_t}$.

Plot a few instances of geometric Brownian motion including drift, building on your R code from the previous exercise.

```r
sigma2<-1
deltax <- rnorm(n = numsteps, sd = sqrt(sigma2/numsteps)) # random changes in position

x <- c(0, cumsum(deltax))
plot(steps, exp(x+2*steps), type = "l", xlab = "Time", ylim = c(0,4), ylab = "Random walk")
lines(steps, exp(2*steps))
for (k in 1:10) {
        sigma2<-1
        deltax <- rnorm(n = numsteps, sd = sqrt(sigma2/numsteps)) # random changes in position

        x <- c(0, cumsum(deltax))
        lines(steps, exp(x+2*steps), type = "l", xlab = "Time", ylim = c(0,4), ylab = "Random walk")

}
```